Master STL - M2 Année 2014/2015

UPMC SORBONNE UNIVERSITÉS

Développement des Applications Réticulaires (DAR)

Novembre 2015

Examen Réparti

---

120 minutes. Tous Documents papier autorisés. Barême à titre indicatif.

Sujet

Le **Turc Mécanique** d'Amazon est une application web réalisant un marché de *crowdsourcing*: elle met en relation des entreprises qui ont des tâches à faire exécuter (en ligne) à des humains et des utilisateurs, qui s'inscrivent, s'engagent à réaliser une tâche, et sont payés par l'entreprise quand la tâche est (correctement et ponctuellement) effectuée.

Des exemples de tâches: traduire des extraits de texte, identifier des images (e.g. comparer deux photographies d'un objet), modérer des forums, écrire des petits scripts, ... Ces tâches doivent être effectués en un temps limité.

Le sujet de cet examen est la réalisation d'un **Turc Mécanique STL** offrant un service similaire. Voici quelques caractéristiques attendues:

- deux types d'utilisateurs: les **Requêteurs** qui postent des tâches à réaliser contre de l'argent, les **Travailleurs** qui choisissent des tâches à effectuer en vue de gagner de l'argent. chaque utilisateur dispose d'un portefeuille sur l'application.
- les tâches et les utilisateurs sont stockés sur **une base de données** (au choix).
- un **ORM** permet au serveur de manipuler facilement utilisateurs et tâches.
- le serveur web, composé de **Servlets**, manipule utilisateurs et tâches comme une **API REST**.
- le client, écrit en **Javascript** (et HTML) permet aux utilisateurs de créer un compte, se connecter en tant que Requêteurs ou Travailleurs.

Les Requêteurs doivent pouvoir (au moins):

- transférer de l'argent depuis un compte en banque vers leur portefeuille,
- poster une nouvelle tâche (ou reposter une tâche mal traitée),
- juger une tâche effectuée (le cas échéant, retirer la tâche correctement effectuée),
- payer un Travailleur (de portefeuille à portefeuille).

Les Travailleurs doivent pouvoir (au moins):

- transférer de l'argent depuis leur portefeuille vers un compte en banque,
- obtenir une liste des tâches disponibles selon des mots-clef,
- s'engager à faire une tâche (un seul travailleur au plus doit être engagé sur la même tâche au même moment),
- poster le résultat d'une tâche,

A Rendre

Le but de l'examen est de réaliser un dossier décrivant l'implémentation de cette application web. Pour chaque point, il convient d'être le plus précis possible (en incluant des extraits de code et des exemples d'utilisation quand c'est possible). Le dossier doit comprendre:

1. un schéma de votre application, détaillant les composants internes coté client et côté serveur, les communications possibles entre les différentes composantes de l'application, annoté avec les différentes technologies utilisées. **(1 pts)**
2. une série de plusieurs *use cases* décrivant textuellement (en prose) des exemples d'utilisations caractéristiques de l'application, **(2 pts)**
3. **DB:** une description de la base de données de l'application, des différentes tables/documents et de leur composition[1] ainsi que quelques exemples d'entrées, **(3pts)**
4. **ORM:** une description des objets Java représentant les données (manipulées par les Servlets) et de l'ORM utilisé (par exemple, un *mapping hibernate*) ainsi que des exemples d'objets correspondant aux exemples d'entrée de la question précédente **(1pt)**,
5. **API:** une description de l'API REST implémentée par l'application, des mot-clefs utilisés, des requêtes possibles, ainsi que plusieurs exemples d'utilisation (requete HTTP, reponse HTTP). **(3 pts)**,
6. **Servlets:** un plan (noms, signatures, rôles) des Servlets présents dans le serveur de l'application, de leurs liens avec les mots-clef de l'API. **(2 pts)**,
7. **Servlet:** le code d'un Servlet caractéristique, au choix **(1 pts)**,
8. **Client:** le plan (descriptions des pages html, noms et rôles des fonctions et scripts) du code JS de la partie client de votre application (pas de description du CSS) ainsi qu'un exemple de page générée (schéma) **(3 pts)**,
9. **Javascript:** le code d'une fonction JS faisant un appel asynchrone à l'API, dans un contexte pertinent **(1 pt)**
10. un plan d'implémentation d'une au plusieurs extensions, au choix ou parmi:
    - un système de réputation des travailleurs et des requêteurs (avec détection des mauvais payeurs et des mauvais travailleurs),
    - des mécanismes d'authentification et sécurité,
    - des liens vers une API de transaction bancaire (imaginaire),
    - prise en charge de l'environnement de réalisation des tâches (affichage d'images, éditeur de texte) par l'application,

[1] Réfléchir, pour chaque tables/documents, à toutes les colonnes/attributs nécessaires.